**[Masini]-Proiect POO**

**Proiect pentru atestat**

**Programare pe obiect**

**[data prezentării]**

|  |  |
| --- | --- |
| **Îndrumător:** | **Student:** |
| **Prof. Carmen Negrea** | **Dicu Manuel (grupa/semigrupa)** |

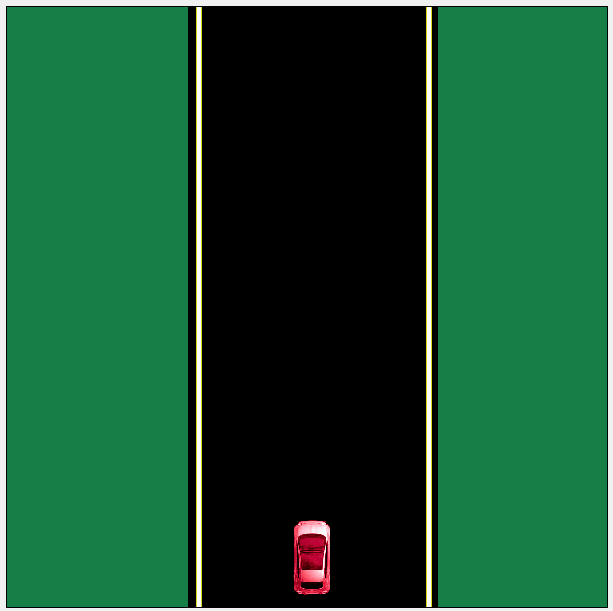
Istoric Versiuni

|  |  |  |  |
| --- | --- | --- | --- |
| **Data** | **Versiune** | **Descriere** | **Autor** |
| 26/03/2014 | 0.01 | Probleme la aparitia copacilor  Probleme la aparitia obiectelor ce necesita miscare  Lipsa variabila pentru daune(daune) | Dicu Manuel |
| 29/03/2014 | 0.1 | Probleme la aparitia copacilor**[rezolvat ulterior]**  Probleme la aparitia obiectelor ce necesita miscare**[rezolvat ulterior]**  Lipsa unui contor fizic pentru scor  Problema de scenariu-masinile parasesc drumul**[rezolvat ulterior]** | Dicu Manuel |
| 21/11/2013 | 0.2-versiune finala | Lipsa probleme  Lipsa unui contor fizic pentru scor**[s-a renuntat datorita complicatiior aparute]** | Dicu Manuel |
| SOON TO COME-urmeaza | 1.0-versiunea finala inbunatatita |  |  |

Scenariu

**Locatie:**

Drum-imagine de tip gif-utilizeaza o suprafata neagra pentru carosabil/marginile nu pot fi depasite e vehicul



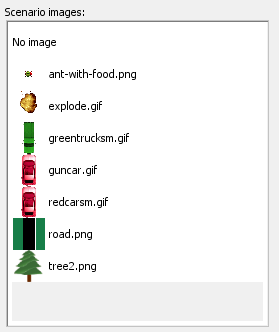
**Obiecte:**

Explozie-imagine de tip gif

Camion-vehicul secundar(greentrucksm.gif)-imagine de tip gif

Guncar/redcarsm-obiect controlat de utilizator-imagini de tip gif

Tree2-copacii de pe marginea carosabilului-imagine de tip gif

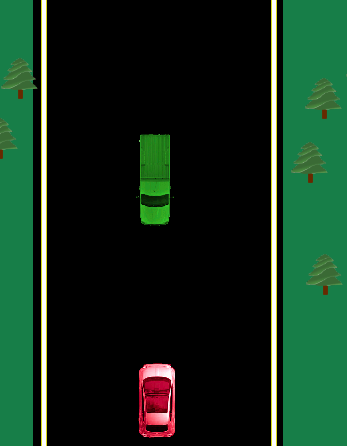


**Actiune:**

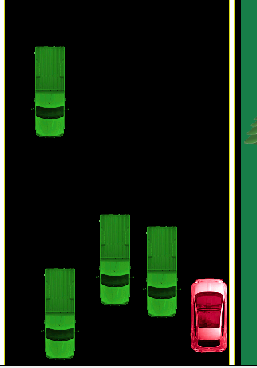
Utilizatorul trebuie sa foloseasca sagetile directionale ale tastaturii pentru a misca vehiculul.

Daca acesta loveste un camion in miscare, o variabila din progarm contorizeaza daunele

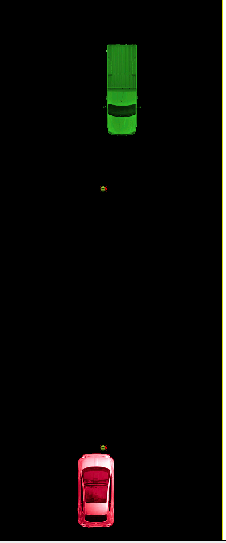
La intalnirea situatiei de mai jos utilizatorul are 2 optiuni



1.Ocolirea obstacolelor



2.Utilizarea armei pentru a distrugerea obstacolelor



Cuprins

Istoric Versiuni 2

Cuprins 3

1.1 Introducere 4

1.1.1 Obiective 4

1.1.2 Scop 4

1.1.3 Definiţii, Tehnici, Acronime şi Abrevieri 4

1.1.4 Tehnologiile utilizate 4

1.2 Cerinţe specifice 4

4 Bibliografie 7

## Introducere

Proiectul este o aplicatie ce are ca scop divertismentul. Prin actiunea si tipul jocului se doreste relaxarea ulizatorului si cultivarea concurentei intre ulizatori.

### Obiective

Obiectivul este realizarea unei aplicatii complet functionale. Apar une glitch-uri datorita programului si erorilor rezultate pana acum.

### Scop

Se doreste ca aplicatia sa fie utilizata pentru divertisement, pentru a descreti fruntile utilizatorilor. Aplicatia este un exemplu perfect de proiect realizat in Greenfoot.

**Despre programarea orientata obiectual**

Programarea orientată pe obiect (Programare Orientată Obiectual) este unul din cei mai importanți pași făcuți în evoluția [limbajelor de programare](http://ro.wikipedia.org/wiki/Limbaj_de_programare) spre o mai puternică abstractizare în implementarea programelor. Ea a apărut din necesitatea exprimării problemei într-un mod mai natural ființei umane. Astfel unitațile care alcătuiesc un program se apropie mai mult de modul nostru de a gândi decât modul de lucru al calculatorului. Până la apariția programării orientate pe obiect, programele erau implementate în [limbaje de programare](http://ro.wikipedia.org/wiki/Limbaj_de_programare) procedurale (C, Pascal) sau în limbaje care nici măcar nu ofereau o modalitate de grupare a instrucțiunilor în unități logice (funcții, proceduri) cum este cazul limbajului de asamblare (asembler). Altfel spus o problemă preluată din natură trebuia fragmentată în repetate rânduri astfel încât să se identifice elementele distincte, implementabile într-un limbaj de programare. O mare problemă a [programării procedurale](http://ro.wikipedia.org/wiki/Programare) era separarea datelor de unitățile care prelucrau datele (subrutinele), ceea ce făcea foarte dificilă extinderea și întreținerea unui program. Astfel s-a pus problema ca aceste două[entități](http://ro.wikipedia.org/wiki/Entitate) (date și subrutine) să fie grupate într-un anumit mod, astfel încât subrutinele să "știe" în permanență ce date prelucrează și, mai mult decât atât, ele să formeze un modul, adică o unitate care separă implementarea de interfață, ceea ce implică posibilitatea refolosirii codului.

* **Abstractizarea** – Este posibilitatea ca un program să ignore unele aspecte ale informației pe care o manipulează, adică posibilitatea de a se concentra asupra esențialului. Fiecare obiect în sistem are rolul unui “actor” abstract, care poate executa acțiuni, își poate modifica și comunica starea și poate comunica cu alte obiecte din sistem fără a dezvălui cum au fost implementate acele facilitați. Procesele, funcțiile sau metodele pot fi de asemenea abstracte, și în acest caz sunt necesare o varietate de tehnici pentru a extinde abstractizarea:
* **Încapsularea** – numită și *ascunderea de informații*: Asigură faptul că obiectele nu pot schimba starea internă a altor obiecte în mod direct (ci doar prin metode puse la dispoziție de obiectul respectiv); doar metodele proprii ale obiectului pot accesa starea acestuia. Fiecare tip de obiect expune o interfață pentru celelalte obiecte care specifică modul cum acele obiecte pot interacționa cu el.
* **Polimorfismul** – Este abilitatea de a procesa obiectele în mod diferit, în funcție de tipul sau de clasa lor. Mai exact, este abilitatea de a redefini metode pentru clasele derivate. De exemplu pentru o clasă Figura putem defini o metodă arie. Dacă Cerc, Dreptunghi, etc. ce vor extinde clasa Figura, acestea pot redefini metoda arie.
* **Moștenirea** – Organizează și facilitează polimorfismul și încapsularea, permițând definirea și crearea unor clase specializate plecând de la clase (generale) deja definite - acestea pot împărtăși (și extinde) comportamentul lor, fără a fi nevoie de a-l redefini. Aceasta se face de obicei prin gruparea obiectelor în clase și prin definirea de clase ca extinderi ale unor clase existente. Conceptul de moștenire permite construirea unor clase noi, care păstrează caracteristicile și comportarea, deci datele și funcțiile membru, de la una sau mai multe clase definite anterior, numite *clase de bază*, fiind posibilă redefinirea sau adăugarea unor date și funcții noi. Se utilizează ideea: ”Anumite obiecte sunt similare, dar în același timp diferite”. O clasă moștenitoare a uneia sau mai multor clase de bază se numește *clasă derivată*. Esența moștenirii constă în posibilitatea refolosirii lucrurilor care funcționează.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* \*

\* RAFT <|-+

\* \* |

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |

|

|

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |

\* \* |

\* RAFT FRIGORIFIC ---+

\* \*

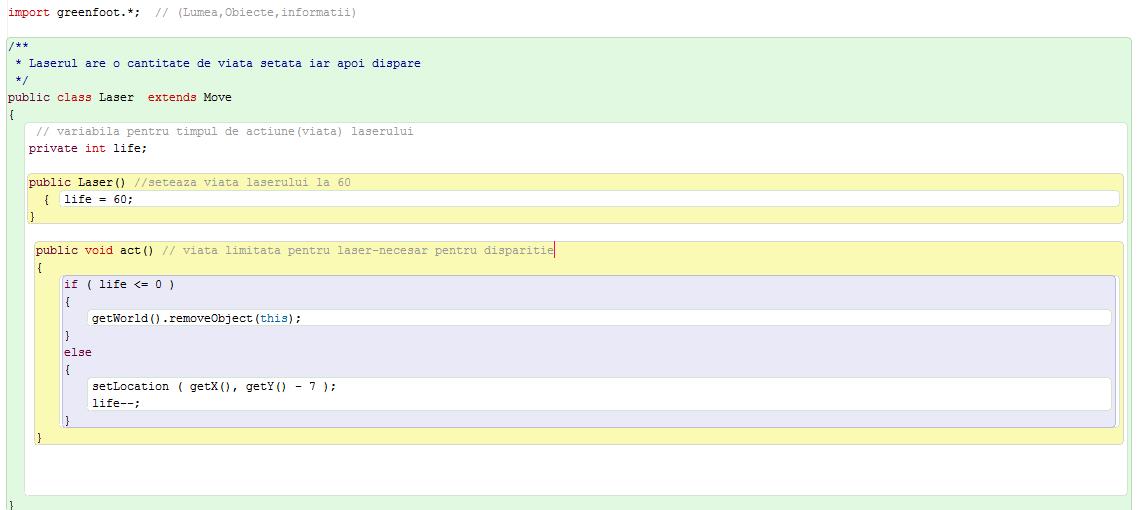
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Clasa Raft Frigorific moștenește clasa Raft

### Definiţii, Tehnici, Acronime şi Abrevieri

Clasa de obiecte=clasa de care obiectele apartin

Tehnicile utilizate:



Varibile pentru timpul de executie

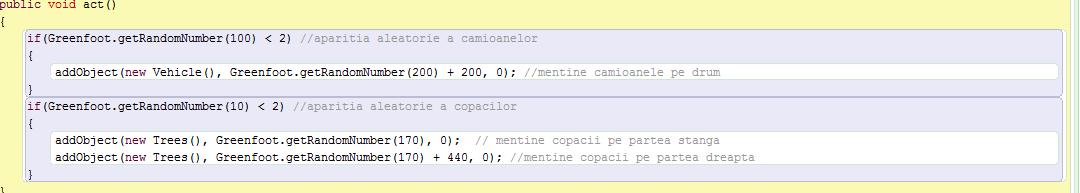
Structuri de tip IF pentru interogarea variabilelor



Intstructiuni de setare si interogare a datelor.

Utilizarea unei structuri pentru libertatea de actiune a obiectelor.

Structuri pentru controlul obiectelor.



Instructiuni permanente ce reglementeaza aparitia si comportamentul obiectelor ce nu pot fi controlate.

### Tehnologiile utilizate

Java+Greenfoot

GREENFOOT

Platforma ideala pentru predarea si invatarea programarii orientate pe obiect de tip JAVA.

Tehnici de programare si efecte vizuale ce faciliteaza invatarea programatorilor aflati la inceput de drum.

Java SE Runtime Mediu conţine maşină virtuală Java, bibliotecile de rulare de clasă, şi aplicarea lansator de Java, care sunt necesare pentru a executa programe scrise în limbajul de programare Java. Acesta nu este un mediu de dezvoltare şi nu conţine instrumente de dezvoltare, cum ar fi compilatoare sau debugger. Pentru instrumente de dezvoltare, a se vedea Java SE Development Kit (JDK (TM)). De instalare JRE declanseaza download Runtime JavaFX. Runtime JavaFX este, de asemenea, disponibile separat, şi nu este parte a JRE.

## Cerinţe specifice

Pentru editarea şi vizualizarea proiectelor construite in Alice 3 sunt necesare două tipuri

de propietai ale calculatorului

Dpdv Software:

-Versiunea 3.x Alice

-Plug-in NetBeans

-Ultimele drivere Java

Dpdv Hardware:

-Procesor (dual-core/quad core)

[Pentium Dual-Core E2140](http://ark.intel.com/products/29738)

* SLA3J (L2)
* SLA93 (M0)
* SLALS (G0)

1.6 GHz1 [MB](http://en.wikipedia.org/wiki/Megabyte)800 MT/s8×1.162–1.312 V65 W[LGA 775](http://en.wikipedia.org/wiki/LGA_775)

* HH80557PG0251M

[Pentium Dual-Core E2160](http://ark.intel.com/products/29739)

* SLA3H (L2)
* SLA8Z (M0)
* SLA9Z (G0)
* SLASX (G0)

1.8 GHz1 MB800 MT/s9×1.162–1.312 V65 WLGA 775

HH80557PG0331M

[Pentium Dual-Core E2180](http://ark.intel.com/products/31733)

* SLA8Y (M0)

2 GHz1 MB800 MT/s10×0.85–1.50 V65 WLGA 775

* HH80557PG0411M

[Pentium Dual-Core E2200](http://ark.intel.com/products/33925)

* SLA8X (M0)

2.2 GHz1 MB800 MT/s11×0.85–1.50 V65 WLGA 77HH80557PG0491M

[Pentium Dual-Core E2220](http://ark.intel.com/products/32430)

* SLA8W (M0)

2.4 GHz1 MB800 MT/s12×0.85–1.50 V65 WLGA 775

* HH80557PG0561M

Sau echivalentul de la AMD.

-Cel putin 2 GB RAM

-Placa video cu memorie Gddr minim 1 GB

# Bibliografie

http://en.wikipedia.org/wiki/List\_of\_Intel\_Pentium\_Dual-Core\_microprocessors

<http://www.alice.org/index.php?page=what_is_alice/what_is_alice>

http://alice3.pbworks.com/w/page/59015092/Alice%203\_1%20FAQ